ACS Synthetic Biology

pubs.acs.org/synthbio Research Article

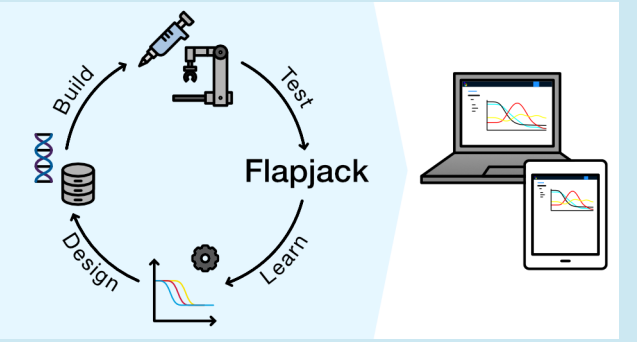
Flapjack: Data Management and Analysis for Genetic Circuit Characterization

Guillermo Yáñez Feliú, Benjamín Earle Gómez, Verner Codoceo Berrocal, Macarena Muñoz Silva, Isaac N. Nuñez, Tamara F. Matute, Anibal Arce Medina, Gonzalo Vidal, Carolus Vitalis, Jonathan Dahlin, Fernán Federici, and Timothy J. Rudge*

Cite This: *ACS Synth. Biol.* 2021, 10, 183–191 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Characterization is fundamental to the design, build, test, learn (DBTL) cycle for engineering synthetic genetic circuits. Components must be described in such a way as to account for their behavior in a range of contexts. Measurements and associated metadata, including part composition, constitute the test phase of the DBTL cycle. These data may consist of measurements of thousands of circuits, measured in hundreds of conditions, in multiple assays potentially performed in different laboratories and using different techniques. In order to inform the learn phase this large volume of data must be filtered, collated, and analyzed. Characterization consists of using this data to parametrize models of component function in different contexts, and combining them to predict behaviors of novel circuits. Tools to store, organize, share, and analyze large volumes of measurement and metadata are therefore essential to linking the test phase to the build and learn phases, closing the loop of the DBTL cycle. Here we present such a system, implemented as a web app with a backend data registry and analysis engine. An interactive frontend provides powerful querying, plotting, and analysis tools, and we provide a REST API and Python package for full integration with external build and learn software. All measurements are associated with circuit part composition *via* SBOL (Synthetic Biology Open Language). We demonstrate our tool by characterizing a range of genetic components and circuits according to composition and context.



KEYWORDS: *synthetic biology, systems biology, data management, characterization, SBOL, web application*

Synthetic biology is an interdisciplinary field that aims to use engineering principles to create novel DNAs with prescribed functions. This process typically follows a design-build-test-learn (DBTL) cycle (Figure 1). DNA design has benefited from developments in syntax,[1−3] part registries,[4−6] Genetic Design Automation (GDA) tools,[7,8] standards for describing and sharing designs,[9] and web-based information systems.[10,11] These tools enable exchange between a collection of open source tools[12] that implement phases of this engineering cycle (Figure 1). For example, simulation tools[13,14] can be used to predict candidate genetic circuits, from which potentially functional circuits are selected. All circuit compositions and part sequences can be designed using the above-mentioned tools to generate files in different formats such as GenBank, FASTA, or SBOL. These designs could be stored and published in SynBioHub[6] or in JBEI-ICE registry,[5] receiving a unique identifier making it possible for the constructs to be accessed by specialized software tools.

The DNA sequence designs generated are then built using modern assembly techniques.[15−18] These techniques are increasingly enabled by robotic automation,[19−22] controlled by open source application programming interfaces (APIs) that can be integrated with different standards. Robotic automation can also be combined with scalable low cost and open source measurement devices that can be run in parallel to generate high-throughput test data.[23,24] These test data feed into the learn phase to estimate parameters of models,[25] train machine learning algorithms,[26] or define design constraints.[27]

Automation is rapidly expanding our capacity to build and test large numbers of genetic circuit designs for applications from microbes[28] to higher organisms such as mammals.[29] However, there remains a need for tools to link these large volumes of test phase data to methods and algorithms to learn new designs, closing the DBTL cycle (Figure 1). Fundamental

Received: October 30, 2020
Published: December 31, 2020

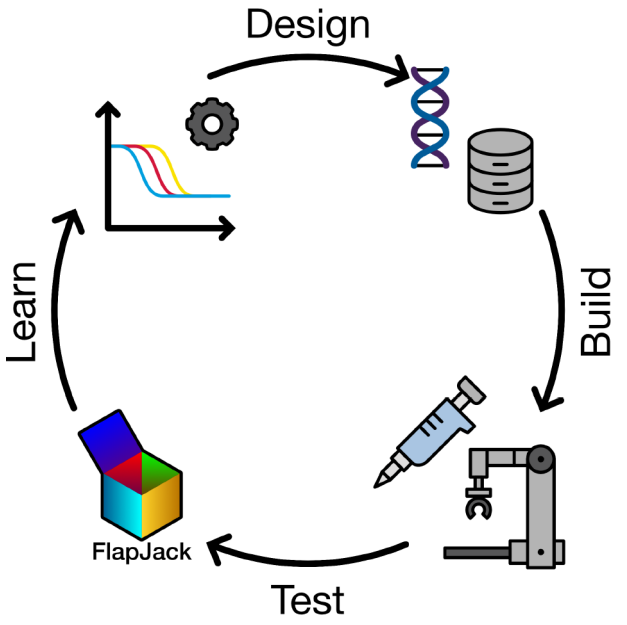


Figure 1. DBTL cycle. The design-build-test-learn (DBTL) cycle can be mostly implemented with existing tools. Flapjack closes the gap between test and learn phases, organizing and analyzing measurements to parametrize or train models.

to the DBTL cycle is characterization of component (part, device, circuit) function based on test data. This data must be collated, filtered, and analyzed to provide the required inputs for the learn phase.[27] Metadata not present in raw data files are also necessary to incorporate details of the DNA design (part composition), experimental conditions (temperature, media, *etc.*), and chassis[30] in which measurements were made. Combining all of this information into a coherent predictive model is the goal of the learn phase.[31−33]

Genetic circuits are dynamical systems driven either by a host cell[31] or a cell-free substrate.[34] To fully characterize their behavior it is therefore necessary to measure and analyze the dynamics of their gene expression. There is currently no best practice for representing the required kinetic measurement data, which is distributed across many files in different formats, stored in different laboratories, institutions, repositories, and databases. While excellent systems biology analysis tools exist for studying gene expression dynamics,[35−37] these operate on single experiments, do not incorporate metadata nor DNA part composition, and do not integrate with searchable registry databases. Neither do existing measurement data registries[11,38−40] reference DNA design part composition *via* exchange standards, such as SBOL.

Here we present a tool which completes the DBTL workflow shown in Figure 1; storing, filtering, visualizing, and analyzing kinetic gene expression data from the test phase in order to provide input to the learn phase.

RESULTS

Data Model. Central to our system is the data model, which defines how the measurement data is organized and labeled (Figure 2, Table 1). The basic unit of data is the *measurement*, which typically consists of readout of fluorescent, luminescent, or colorimetric reporter level,[41,42] and estimates of biomass or density. Each measurement refers to a particular *sample* at a specific time, and each sample is measured at multiple time points, possibly reading multiple reporter levels. Each sample is measured in a specific *assay*, which might correspond for example to a kinetic plate-reader experiment. Finally, sets of assays are grouped into a *study* representing a project with a common purpose. This hierarchy represents typical lab workflows and allows users to easily filter data intuitively, providing a platform for data exploration.

However, in order to fully characterize the behavior of genetic circuits and their components one must consider the context[30,43] in which measurements were recorded. This context is given by metadata which are stored in our database and related to each measurement. For example, to account for metabolic changes, the chassis strain used, the media in which it grew, and the assay temperature must be considered during data analysis. External signals in the form of inducer chemicals must also be incorporated to parametrize models of gene regulation. Most importantly, using the stored SynBioHub URI of genetic circuits and connecting to SynBioHubs[6] *via* PySBOL,[44] we can analyze the composition of the DNA and relate the context of components to their effect on circuit behavior. This combination of measurement data and contextual metadata enables sophisticated analysis and modeling[31] of genetic circuits that can be used to learn increasingly complex designs.[45,46] Identifying DNAs by their SynBioHub URI means that Flapjack effectively integrates measurement data, its analysis, and its metadata to the exact sequence of parts from which the circuit DNA is composed.

Web Interface and API. The architecture of Flapjack is shown in Figure 3 and it is composed of a web interface frontend and a backend application programming interface (API). We provide a comprehensive web interface that first allows users to create a new study, or add assays to an existing study, by uploading data files and inputting additional metadata interactively *via* the web app. Currently two sources of data are supported, microplate readers and FluoPi fluorescence imaging stations,[23] although it is straightforward to write other parsers. In the case of multiwell microplate data, required metadata that varies between wells is specified in an Excel file containing various sheets, with each cell a well in the plate. Raw data is also contained in a sheet of this file. In the case of FluoPi, data is uploaded as a JSON file output by the FluoPi analysis software, and metadata is supplied *via* the user interface.

On the browse page, the user may browse and search available studies, assays and vectors. On a separate tab users may also search and browse a table of assays, viewing a short description of each. The vector tab provides an optional SynBioHub URI which links to a SynBioHub instance[47] where it is possible to download the SBOL design file[6] and SBOL circuit diagram,[48] as well as inspect part composition. Sharing and privacy of data is controlled at the study level and consists of three types: private, shared with specific users, and public (visible to all users). In a typical workflow, we envisage a team of researchers, each generating, analyzing, and visualizing their own data. When a team member has useful results they share those data with other team members. Later if the collaboration generates a published outcome, the results can be made public.

On the view page (Figure 4) Flapjack provides a tabbed environment in which to explore and analyze data. Users construct a query by selecting filters for study, assay, media, strain, vector, and signal. For example, one might query all data for a given vector, irrespective of the study in which it was performed. This data can then be visualized interactively,

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191

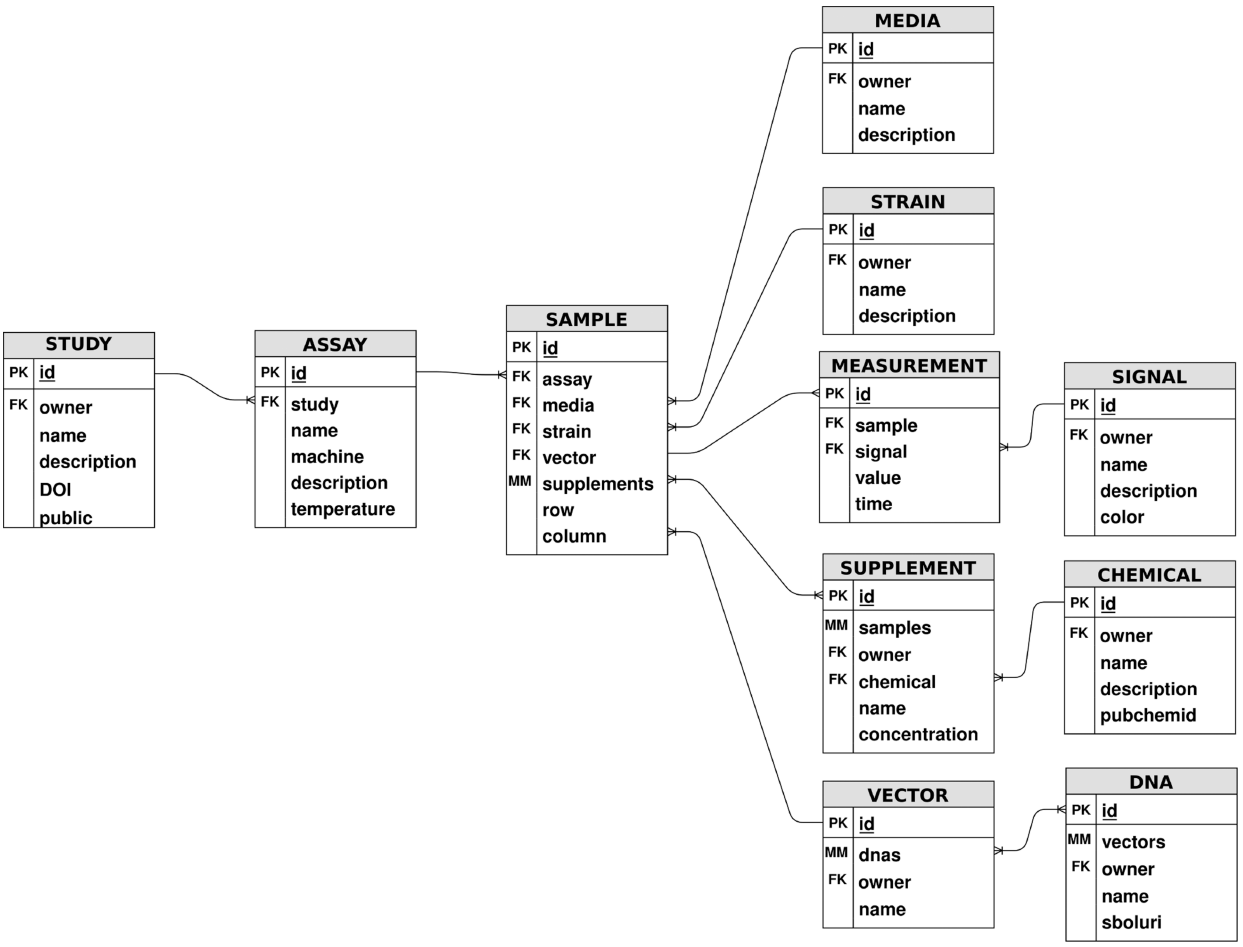


Figure 2. Flapjack's data model. The basic unit of data is the *measurement*, which is the readout of a reporter or biomass *signal* in time. A *measurement* refers to a specific *sample*, which can be a well in a plate-reader experiment or a colony in a Petri dish time-lapse. *Samples* are grouped in a specific *assay*, which corresponds to a single kinetic procedure. *Assays* belong to a particular *study* that aims to answer a scientific question or project. Along with *signal* data, each *measurement* is described in terms of its context through experimental metadata, mainly composed of the *media* substrate, the host *strain*, the *DNA* which the host is transformed with, and inducer *chemicals* to study stress or other regulation behaviors of the system. It is important to mention that some metadata could not be present, such as *strain* in Cell-free *assays* or *DNA* in control *samples*, to name a few. PK is primary key, FK is foreign key, and MM is many-to-many relation.

Table 1. Objects That Constitute the Flapjack Data Model

object	description
Study	A project, for example a paper or report, that corresponds to a certain question a researcher wants to address.
Assay	Measurement experiments, including replicates and varying experimental conditions, performed to explore different aspects of the study.
Sample	Corresponds to the basic unit that is subject to measurement, for example a colony or a well in a microplate.
Measurement	The value of the raw measurement recorded for a particular sample during an assay at a particular time.
Vector	Describes the synthetic DNAs encoding a genetic circuit, including links to part composition and sequence *via* the corresponding SynBioHub URIs.
Media	Composition of the substrate that drives the genetic circuit, media in the case of live cell assays, or extract for cell-free experiments.
Strain	The chassis organism, if any, hosting the genetic circuit.
Supplement	Any supplementary chemicals that interact with components of the genetic circuit.
Signal	The subject of measurements, for example a fluorescence channel with given filter bandwidths.

plotting each measurement as a function of time. Plots are grouped into subplots, and given different line colors according to user-selected labels. For example the user might compare two studies by creating two tabs in the user interface. Within each tab the user could also select to group the data into subplots according to the vector in each sample. Finally, it is possible to group the lines into colors according to the signal, for example to differentiate fluorescence channels. Data can also be summarized by plotting mean and standard deviation and/or normalized for comparison. The plots generated by the frontend can be downloaded as print quality PNG files with specified dimensions and font size, or as JSON encoded figure objects for formatting with Plotly.[49] Plots are stored in the browser's local storage for persistence while sessions are active.

Having constructed a query, the user can also perform various analyses, plotting the results according to the same groupings. The user provides input parameters to a range of functions that transform the selected data time series. For example, the *mean expression* summarizes a kinetic experiment by the average reporter level, and results in a set of bar graphs according to the specified grouping options. More complex analyses include calculation of expression (or synthesis) rates, induction curves, velocity profiles, and induction kymographs

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191

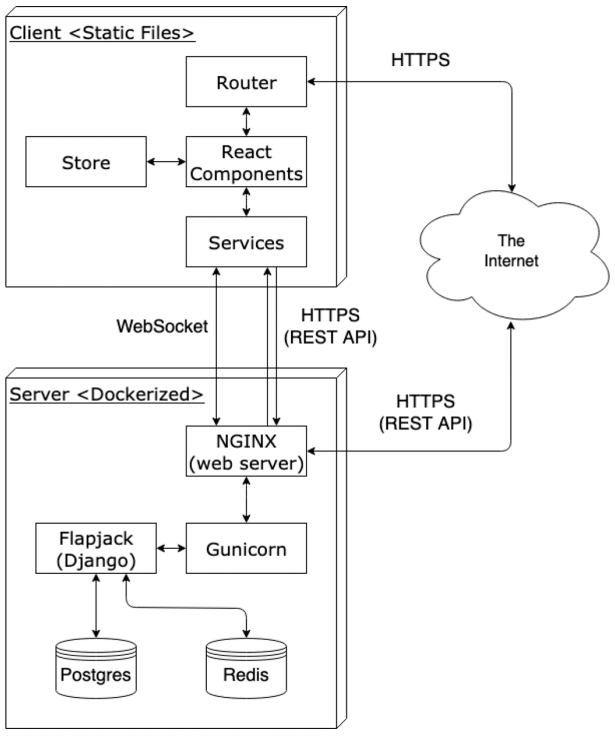


Figure 3. Flapjack's architecture. Flapjack is a full stack web application developed as a Django REST API and a React web client, built on a PostgreSQL database and deployed in Docker containers for ease of use and portability. The frontend-backend connection is made *via* the REST API and WebSockets when persistence is needed, both using JSON (JavaScript Object Notation) web tokens for authentication to access restricted end points. The backend makes use of NGINX to reverse proxy to a Gunicorn application, Redis database to store data related to WebSockets and data science libraries to process data and perform analysis such as NumPy, Pandas, and SciPy. The frontend uses D3.js library *via* Plotly to visualize data, Redux and Redux Persist for global state management and data persistence. We also provide a Python package to access progammatically to the REST API. In order to facilitate interoperability with external tools, all data exchange is made using JSON.

(see Supporting Information for examples). Growth can also be considered, using the same tools to extract mean or maximum biomass or growth rate by analyzing measurements that correspond to biomass estimates. A comprehensive set of tutorials for the capabilities and use of the web interface is provided on our wiki (https://github.com/synbiouc/flapjack_frontend/wiki; see also Supporting Information).

Analysis of gene expression measurement data requires removal of background signal from the growth medium, and from the cells themselves, for example, autofluorescence. In order to do this, experiments should be designed with control samples containing only growth media, and samples containing untransformed host cells for *in vivo* assays. Flapjack automatically detects control samples, those without cells (media background) and those without DNA (untransformed cells), and computes and subtracts signal background where necessary. The time-varying background signals are computed as the mean signal value at each time point. The standard deviation can be used to set a threshold below which data points may be considered background and optionally removed from analysis.

Queries automatically aggregate data from multiple studies and assays, picking out particular samples according to the specified metadata. All querying, plotting and analysis functions and resulting figures and data may also be accessed *via* our Python package (https://github.com/synbiouc/flapjack). *Via* the API, this package interfaces Flapjack directly with Pandas[50] and thus to the Numpy/SciPy stack.[51,52] Pandas is a powerful data analysis library that enables filtering, merging, pivoting among other operations on tabulated data or dataframes. Columns in these dataframes, such as time series of measurements, are represented as Numpy arrays, meaning that a broad range of signal processing, statistical analysis, and visualization techniques may be applied to them. Figures downloaded as JSON may be formatted and added to (*e.g.*, annotated) using Plotly[49] *via* Python[53] or JavaScript. Our API uses JSON making it simple to interface with other software and collaborate on data analysis outside of Flapjack.

Design-Build-Test-Learn Cycle. We used Flapjack to complete the DBTL cycle (Figure 1), which can be summarized as follows. First, in the design phase circuits were composed using SBOLDesigner[8] and uploaded to our SynBioHub instance.[47] The build phase, DNA assembly, was performed manually but could be automated using open source APIs that can be linked to genetic designs.[16,22] Test phase data was generated using various kinetic gene expression measurement techniques, which can be automated to varying degrees.[21−24] Test data and metadata was then uploaded to Flapjack, allowing visualization and analysis of the synthetic genetic circuits measured under various conditions. This data was then used to quantitatively characterize circuit behavior, allowing estimation of parameters for models to predict new designs, which could be simulated using custom software or existing simulation tools.[13,14] These simulations can predict novel circuit behavior, constituting the learn phase.[31] Furthermore, Flapjack provides *tidy* analysis data with characteristic features linked to metadata labels that can directly feed into machine learning and statistical analysis toolkits such as scikit-learn[54] and TensorFlow,[55] which enable supervised and unsupervised learning, including deep neural networks.

DISCUSSION

Synthetic biology aims to apply the engineering principles of design, build, test, and learn to biological systems, either to mimic existing functions in nature or to create novel genetic components, networks, and pathways. Recent developments in design tools,[56] DNA assembly,[15] DNA composition registries,[5,6] and robotics automation are enabling not only DNA assembly in bacteria[22] but also for plants[57] and yeast,[58] cell-free cloning of custom DNA libraries in microfluidic devices,[59] and even chemical production.[60] Further, combined with this large-scale circuit assembly, developments in automated measurement assays[23,24] will enable the generation of large volumes of data *via* high-throughput experiments. These data must also be linked to metadata or labels that give context to the measurement,[61,62] such as growth conditions, host strain, and supplementary inducer chemical concentrations, as well as relating them to the circuit part composition. This scenario creates the need for tools that apply the DBTL principle, organizing kinetic gene expression data from high-throughput assays and characterizing the behavior of the biological systems under study, linking these large volumes of test phase data to the learn phase in order to complete the engineering cycle.

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191

Figure 4. The web interface of Flapjack. The view page allows users to query data stored in the database and perform a variety of analysis. Queries and analysis can then be visualized interactively in plots grouped into subplots and given different line colors, both according to user-selected labels (*i.e., assay, vector, signal, etc.*). Also, data can be summarized by plotting mean and standard deviation and/or normalized for comparison. These plots can be organized in different tabs and stored in the browser's local storage while the session is active, thus users can compare between different querying filters. Finally, users can download the generated plots as PNG files or as JSON encoded figures for formatting with Plotly and use them for publication purposes.

We developed a standardized data model that enables scalable data storage, analysis, visualization, and parameter estimation, enabling collation and sharing of gene expression data from diverse sources and different formats. Our data model links measurements to metadata describing the conditions under which they were taken, and to the design of the genetic circuit under study through connecting directly to an existing ecosystem of standards and software tools. We demonstrated that Flapjack closes the gap between the DBTL cycle build and learn phases by characterizing a variety of genetic circuits with data from different protocols and measurement devices. Using our system we were able to characterize important synthetic biology technologies such as CRISPRi transcription regulation,[63] cell-free protein synthesis,[64] and image-based measurements of genetic circuit dynamics.

Flapjack uses state-of-the-art systems biology methods[36] to extract robust measures of genetic circuit dynamics. We provide an engine to generate dynamic analyzes such as expression rate and cell-free reaction velocity profiles and induction kymographs, as well as summary measures such as mean/max expression levels and induction curves. While not presented in these results, Flapjack also generates heatmaps for the characterization of two inducer systems (such as NOR or AND gates) and ratiometric analyzes that measure promoter activity relative to a reference promoter.[65,66] All analyzes are related to genetic circuit context *via* metadata, including cellular context related to operational conditions such as media composition and host strain, and compositional context due to circuit part composition *via* the SBOL exchange standard[9] and SynBioHub.[6]

We facilitate the synthetic biology workflow by the use of a modern architecture which connects to an ecosystem of external tools. The platform makes use of the latest technologies in web development. Its microservices architecture *via* the use of Docker virtualizations makes installation and deployment easy for administrators and developers. Flapjack's backend engine was developed using Django and Django Rest Framework, thus supporting many Python tools especially for data analysis. Despite the effectiveness of this architecture, user experience is key to widespread adoption of any tool. We used frameworks such as React, Redux, and D3.js to create a comprehensive, intuitive and fast user interface that exposes our powerful storage, analysis, and visualization engines in an efficient and user-friendly fashion. This is essential to lower entry barriers for nonexpert users.

Standards are essential to facilitate the automation and exploration of large design spaces[62] and Flapjack provides a template to standardize storage, exchange, and collaborative sharing of kinetic gene expression data. The engineering cycle

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191

relies heavily on characterization of genetic parts, devices, and systems,[67,68] creating a need for analysis tools that integrate experimental data with DNA sequences. As circuits get larger, the process of constructing and testing them becomes more difficult. Modeling plays a key role in the characterization of biological systems and with the use of mathematical and computational techniques researchers could optimize their designs rather than testing them in the lab, reducing guesswork and trial-and-error. Model-based design techniques in synthetic biology are facing a bottleneck in data management and analysis that Flapjack overcomes.[69] We envisage that software implementing such techniques would connect to Flapjack *via* the API to extract the necessary data to parametrize its specific models, which could leverage standards such as SBML.

While our system uses a simple JSON format readable by many software packages, in future, tighter integration with SBOL and SBML will create more efficient and accessible workflows. Robotic automation APIs[22] and lab inventory management systems could be connected to Flapjack providing a direct link between data and physical DNA. With the advent of large-scale DNA assembly techniques and automation of laboratory instruments the amount of data synthetic biologists will face in future will be massive. Leveraging such data will be essential to develop machine learning based design algorithms and assembly automation protocols, which will allow important advances in synthetic biology.[33,70] Biofoundry alliances[71] have already adopted DNA assembly standards[62] for high-throughput circuit construction and will now face the data management and analysis tasks that Flapjack addresses in order to drive new economic sectors producing pharmaceuticals, biomaterials, clean fuels, and sustainable food, promoting a greener economy.

METHODS

Web Application and API. The web application was developed using a microservices architecture with Docker containers virtualization system. The backend is written in Python[53] using the Django framework as a base for the API and WebSockets architecture. The Django Rest Framework library is used to build a powerful and flexible REST API, while Pandas,[50] NumPy,[51] and SciPy[52] libraries are used to process the data before sending it to the frontend. Using common data structures such as Numpy arrays means that the backend can be easily modified to implement different analyses and interface with other analysis libraries, as we do with the WellFARE systems biology package.[36] The API provided by the backend, in addition to serve the frontend with the services it needs, can be used to interact with Flapjack for more sophisticated analysis by connecting to existing libraries and software[6,50,54,55,72] via a provided Python package (tutorials for the package are also available; https://github.com/SynBioUC/flapjack/tree/master/notebooks).

The frontend interface is written in JavaScript, using the React library[73] for an interactive design, the D3[74] visualization library *via* Plotly[49] and Redux for global state management and data persistence. This application is intended for convenient data exploration and basic analysis, and can also be deployed using Docker[75] for ease of setup. Access to user data in the frontend is managed *via* JSON web tokens, where the refresh token is stored in the browser's local storage *via* Redux Persist, and is used upon site load to obtain the user's access token. In the first instance, the user (owner) who uploads the data is the only one who has access to it. We have implemented a data sharing system that considers giving access to particular users or making studies public. Thus, we ensure that only those authorized by the owner have access to the information.

The connection between frontend and backend is made *via* a REST API provided by the backend and WebSocket connections for functionalities that require a persistent connection. Both the API and WebSockets function with authentication *via* JSON web tokens, which are required to access restricted end points. JSON is used for all data exchange, maximizing flexibility and interoperability with external tools.

In order to achieve performance and security in the backend, an NGINX server is used to reverse proxy to a Gunicorn application that serves the Django Application, and to process the requests. All persistent data is stored in a PostgreSQL database, while a Redis database is used to store data related to WebSocket connections and messages. The backend is deployed using Docker[75] for easy installation and portability.

Details about the installation and configuration of the Flapjack web app can be found at github.com/synbiouc/flapjack_api, github.com/synbiouc/flapjack_frontend, and the Python package may be installed following the instructions at github.com/synbiouc/flapjack. A public instance of Flapjack with all data presented in this study can be found at flapjack.rudge-lab.org.

Experimental Procedures. DNA assembly was carried out using Golden Gate assembly[76] and the method of Gibson.[77] Measurement assays were performed using a Synergy HTX microplate reader and a FluoPi imaging station.[23] Cell-free extracts were prepared as described in the Supporting Information. Full details of all experimental procedures can be found in the Supporting Information. All data plots presented were generated using the Flapjack frontend or API.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acssynbio.0c00554.

Flapjack backend documentation (PDF)

Flapjack frontend documentation (PDF)

Tables, figures, and detailed methods (PDF)

AUTHOR INFORMATION

Corresponding Author

Timothy J. Rudge − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine and Department of Chemical and Bioprocess Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile;* orcid.org/0000-0001-9446-9958; Email: trudge@uc.cl

Authors

Guillermo Yáñez Feliú − *Department of Chemical and Bioprocess Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile;* orcid.org/0000-0002-3703-7961

Benjamín Earle Gómez − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile*

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191

Verner Codoceo Berrocal − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile*

Macarena Muñoz Silva − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile*

Isaac N. Nuñez − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine and Department of Chemical and Bioprocess Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile; ANID − Millennium Science Initiative Program − Millennium Institute for Integrative Biology (iBio), Pontificia Universidad Católica de Chile, Santiago 8330005, Chile*

Tamara F. Matute − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine and Department of Chemical and Bioprocess Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile; ANID − Millennium Science Initiative Program − Millennium Institute for Integrative Biology (iBio), Pontificia Universidad Católica de Chile, Santiago 8330005, Chile*

Anibal Arce Medina − *Departamento de Genética Molecular y Microbiología, Facultad de Ciencias Biológicas, Pontificia Universidad Católica de Chile, Santiago 8330005, Chile; ANID − Millennium Science Initiative Program − Millennium Institute for Integrative Biology (iBio), Pontificia Universidad Católica de Chile, Santiago 8330005, Chile*

Gonzalo Vidal − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile*

Carolus Vitalis − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile*

Jonathan Dahlin − *The Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, 2800 Lyngby, Denmark*

Fernán Federici − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile; ANID − Millennium Science Initiative Program − Millennium Institute for Integrative Biology (iBio) and FONDAP, Center for Genome Regulation, Pontificia Universidad Católica de Chile, Santiago 8330005, Chile*

Complete contact information is available at: https://pubs.acs.org/10.1021/acssynbio.0c00554

Author Contributions

TJR and GYF designed the study. GYF, TJR, BEG, and VCB wrote the software. All authors contributed to writing the manuscript. MMS, INN, TFM, AAM, and JD performed the experiments.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

GYF was supported by Beca Ayudante Doctorando scholarship from the Department of Chemical and Bioprocess Engineering, Pontificia Universidad Católica de Chile. GV was supported by a scholarship from the Institute for Biological and Medical Engineering, Pontificia Universidad Católica de Chile. TJR, GYF, GV, MMS were supported by ANID PIA Anillo ACT192015. FF, AA, INN, TFM were funded by ANID − Millennium Science Initiative Program − ICN17_022, Fondo de Desarrollo de Areas Prioritarias − Center for Genome Regulation (ANID/FONDAP/15090007) and CONICYT Fondecyt Iniciacion 11140776. INN was supported by VRI grant from Escuela de Graduados de la Vicerrectoría de Investigación UC. AAM was supported by Beca Doctorado Nacional Conicyt 21140714 − Millennium Science Initiative Program − Millennium Institute for Integrative Biology (iBio). TJR, GYF, GV, MMS were supported by CONICYT Fondecyt Iniciacion 11161046. The authors thank Alvaro Olivera for his support with the iGEM kit and Kevin Simpson for his help with preliminary experiments and results.

■ REFERENCES

(1) Registry of Standard Biological Parts. http://parts.igem.org/Main_Page (Accessed on 2020−12−24).

(2) Weber, E., Engler, C., Gruetzner, R., Werner, S., and Marillonnet, S. (2011) A modular cloning system for standardized assembly of multigene constructs. *PLoS One 6*, 1−11.

(3) Patron, N. J., et al. (2015) Standards for plant synthetic biology: a common syntax for exchange of DNA parts. *New Phytol. 208*, 13−19.

(4) iGem Parts Repository. https://igem.org/Registry (Accessed on 2020−12−24).

(5) Inventory of Composable Elements (ICE). https://public-registry.jbei.org (Accessed on 2020−12−24).

(6) McLaughlin, J. A., Myers, C. J., Zundel, Z., Mlslrll, G., Zhang, M., Ofiteru, I. D., Goñi-Moreno, A., and Wipat, A. (2018) SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. *ACS Synth. Biol. 7*, 682−688.

(7) Chandran, D., Bergmann, F. T., and Sauro, H. M. (2009) TinkerCell: Modular CAD tool for synthetic biology. *J. Biol. Eng. 3*, 1−17.

(8) Zhang, M., McLaughlin, J. A., Wipat, A., and Myers, C. J. (2017) SBOLDesigner 2: An Intuitive Tool for Structural Genetic Design. *ACS Synth. Biol. 6*, 1150−1160.

(9) Galdzicki, M., et al. (2014) The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol. 32*, 545−550.

(10) Bultelle, M., De Murieta, I. S., and Kitney, R. (2016) Introducing SynBIS -The synthetic biology information system. *IET Conference Publications 2016*, 2−3.

(11) Morrell, W. C., et al. (2017) The Experiment Data Depot: A Web-Based Software Tool for Biological Experimental Data Storage, Sharing, and Visualization. *ACS Synth. Biol. 6*, 2248−2259.

(12) Urquiza-García, U., Zieliński, T., and Millar, A. J. (2019) Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs. *Synth. Biol. 4*, 1−8.

(13) Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. (2011) Eugene − A Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems. *PLoS One 6*, 1−12.

(14) Watanabe, L., Nguyen, T., Zhang, M., Zundel, Z., Zhang, Z., Madsen, C., Roehner, N., and Myers, C. J. (2019) IBioSim 3: A Tool for Model-Based Genetic Circuit Design. *ACS Synth. Biol. 8*, 1560−1563.

(15) Casini, A., Storch, M., Baldwin, G. S., and Ellis, T. (2015) Bricks and blueprints: Methods and standards for DNA assembly. *Nat. Rev. Mol. Cell Biol. 16*, 568−576.

(16) Storch, M., Casini, A., Mackrow, B., Fleming, T., Trewhitt, H., Ellis, T., and Baldwin, G. S. (2015) BASIC: A New Biopart Assembly

Standard for Idempotent Cloning Provides Accurate, Single-Tier DNA Assembly for Synthetic Biology. *ACS Synth. Biol. 4*, 781−787.
(17) Rice, E. S., and Green, R. E. (2019) New Approaches for Genome Assembly and Scaffolding. *Annu. Rev. Anim. Biosci. 7*, 17−40.
(18) Pollak, B., Matute, T., Nuñez, I., Cerda, A., Lopez, C., Vargas, V., Kan, A., Bielinski, V., von Dassow, P., Dupont, C. L., and Federici, F. (2020) Universal Loop assembly (uLoop): open, efficient, and cross-kingdom DNA fabrication. *Synth. Biol. 5*, 1−13.
(19) Opentrons website. https://www.opentrons.com (Accessed on 2020−12−24).
(20) Ortiz, L., Pavan, M., McCarthy, L., Timmons, J., and Densmore, D. M. (2017) Automated robotic liquid handling assembly of modular DNA devices. *J. Visualized Exp. 130*, 1−7.
(21) Walsh, D. I., Pavan, M., Ortiz, L., Wick, S., Bobrow, J., Guido, N. J., Leinicke, S., Fu, D., Pandit, S., Qin, L., Carr, P. A., and Densmore, D. (2019) Standardizing Automated DNA Assembly: Best Practices, Metrics, and Protocols Using Robots. *SLAS Technology 24*, 282−290.
(22) Storch, M., Haines, M., and Baldwin, G. (2020) DNA-BOT: A low-cost, automated DNA assembly platform for synthetic biology. *Synth. Biol. 5*, 1−7.
(23) Nuñez, I., Matute, T., Herrera, R., Keymer, J., Marzullo, T., Rudge, T., and Federici, F. (2017) Low cost and open source multi-fluorescence imaging system for teaching and research in biology and bioengineering. *PLoS One 12*, 1−21.
(24) French, S., Coutts, B. E., and Brown, E. D. (2018) Open-Source High-Throughput Phenomics of Bacterial Promoter-Reporter Strains. *Cell Systems 7*, 339−346.
(25) Tamsir, A., Tabor, J. J., and Voigt, C. A. (2011) Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature 469*, 212−215.
(26) Borkowski, O., Koch, M., Zettor, A., Pandi, A., Batista, A. C., Faulon, J.-l., and Soudier, P. (2020) Large scale active-learning-guided exploration for in vitro protein production optimization. *Nat. Commun. 11*, 1−8.
(27) Nielsen, A. A., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016) Genetic circuit design automation. *Science 352*, 53−64.
(28) Ren, J., Lee, J., and Na, D. (2020) Recent advances in genetic engineering tools based on synthetic biology. *J. Microbiol. 58*, 1−10.
(29) Kojima, R., and Fussenegger, M. (2019) Synthetic Biology: Engineering Mammalian Cells To Control Cell-to-Cell Communication at Will. *ChemBioChem 20*, 994−1002.
(30) Borkowski, O., Ceroni, F., Stan, G. B., and Ellis, T. (2016) Overloaded and stressed: whole-cell considerations for bacterial synthetic biology. *Curr. Opin. Microbiol. 33*, 123−130.
(31) Liao, C., Blanchard, A. E., and Lu, T. (2017) An integrative circuit-host modelling framework for predicting synthetic gene network behaviours. *Nature Microbiology 2*, 1658−1666.
(32) Camacho, D. M., Collins, K. M., Powers, R. K., Costello, J. C., and Collins, J. J. (2018) Next-Generation Machine Learning for Biological Networks. *Cell 173*, 1581−1592.
(33) Carbonell, P., Radivojevic, T., and García Martín, H. (2019) Opportunities at the Intersection of Synthetic Biology, Machine Learning, and Automation. *ACS Synth. Biol. 8*, 1474−1477.
(34) Kopniczky, M. B., Canavan, C., McClymont, D. W., Crone, M. A., Suckling, L., Goetzmann, B., Siciliano, V., MacDonald, J. T., Jensen, K., and Freemont, P. S. (2020) Cell-Free Protein Synthesis as a Prototyping Platform for Mammalian Synthetic Biology. *ACS Synth. Biol. 9*, 144−156.
(35) Boyer, F., Besson, F. B., Baptist, G., Izard, J., Pinel, C., Ropers, D., Geiselmann, J., and de Jong, H. (2010) WellReader: A MATLAB program for the analysis of fluorescence and luminescence reporter gene data. *Bioinformatics 26*, 1262−1263.
(36) Zulkower, V., Page, M., Ropers, D., Geiselmann, J., and De Jong, H. (2015) Robust reconstruction of gene expression profiles from reporter gene data using linear inversion. *Bioinformatics 31*, i71−i79.
(37) De Jong, H., Casagranda, S., Giordano, N., Cinquemani, E., Ropers, D., Geiselmann, J., and Gouzé, J. L. (2017) Mathematical modelling of microbes: Metabolism, gene expression and growth. *J. R. Soc., Interface 14*, 1−14.
(38) Huynh, L., and Tagkopoulos, I. (2016) A Parts Database with Consensus Parameter Estimation for Synthetic Circuit Design. *ACS Synth. Biol. 5*, 1412−1420.
(39) Martin, Y., Page, M., Blanchet, C., and De Jong, H. (2019) WellInverter: A web application for the analysis of fluorescent reporter gene data. *BMC Bioinf. 20*, 1−18.
(40) Datanator website. https://datanator.info (Accessed on 2020−12−24).
(41) Slomovic, S., Pardee, K., and Collins, J. J. (2015) Synthetic biology devices for in vitro and in vivo diagnostics. *Proc. Natl. Acad. Sci. U. S. A. 112*, 14429−14435.
(42) Gilad, A. A., and Shapiro, M. G. (2017) Molecular Imaging in Synthetic Biology, and Synthetic Biology in Molecular Imaging. *Molecular Imaging and Biology 19*, 373−378.
(43) Yeung, E., Dy, A. J., Martin, K. B., Ng, A. H., Del Vecchio, D., Beck, J. L., Collins, J. J., and Murray, R. M. (2017) Biophysical Constraints Arising from Compositional Context in Synthetic Gene Networks. *Cell Systems 5*, 11−24.
(44) Bartley, B. A., Choi, K., Samineni, M., Zundel, Z., Nguyen, T., Myers, C. J., and Sauro, H. M. (2019) pySBOL: A Python Package for Genetic Design Automation and Standardization. *ACS Synth. Biol. 8*, 1515−1518.
(45) Purnick, P. E., and Weiss, R. (2009) The second wave of synthetic biology: From modules to systems. *Nat. Rev. Mol. Cell Biol. 10*, 410−422.
(46) Grunberg, T. W., and Del Vecchio, D. (2020) Modular Analysis and Design of Biological Circuits. *Curr. Opin. Biotechnol. 63*, 41−47.
(47) SynBioHubUC. http://synbiohub.rudge-lab.org/ (Accessed on 2020−12−24).
(48) McLaughlin, J. A., Pocock, M., Mlslrll, G., Madsen, C., and Wipat, A. (2016) VisBOL: Web-Based Tools for Synthetic Biology Design Visualization. *ACS Synth. Biol. 5*, 874−876.
(49) Collaborative data science. 2015; https://plot.ly (Accessed on 2020−12−24).
(50) McKinney, W. (2010) Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, pp 56−61.
(51) Harris, C. R., et al. (2020) Array programming with NumPy. *Nature 585*, 357−362.
(52) Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Walt, S. J. V. D., Brett, M., Wilson, J., Millman, K. J., et al. (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods 17*, 261−272.
(53) Van Rossum, G., and Drake, F. L. (2009) *Python 3 Reference Manual*, CreateSpace: Scotts Valley, CA.
(54) Pedregosa, F., et al. (2011) Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res. 12*, 2825−2830.
(55) Abadi, M., et al. (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; https://www.tensorflow.org/ (Accessed on 2020−12−24).
(56) Chen, Y., Zhang, S., Young, E. M., Jones, T. S., Densmore, D., and Voigt, C. A. (2020) Genetic circuit design automation for yeast. *Nat. Microbiol. 5*, 79−81.
(57) Mann, D. G. J., Bevan, S. A., Harvey, A. J., and Leffert-sorenson, R. A. (2019) The Use of an Automated Platform to Assemble Multigenic Constructs for Plant Transformation. *Methods Mol. Biol. 1864*, 19−35.
(58) Rajakumar, P. D., Gowers, G. O., Suckling, L., Foster, A., Ellis, T., Kitney, R. I., McClymont, D. W., and Freemont, P. S. (2019) Rapid Prototyping Platform for Saccharomyces cerevisiae Using Computer-Aided Genetic Design Enabled by Parallel Software and Workcell Platform Development. *SLAS Technology 24*, 291−297.
(59) Ben Yehezkel, T., Rival, A., Raz, O., Cohen, R., Marx, Z., Camara, M., Dubern, J. F., Koch, B., Heeb, S., Krasnogor, N., Delattre, C., and Shapiro, E. (2016) Synthesis and cell-free cloning of DNA

libraries using programmable microfluidics. *Nucleic Acids Res. 44*, 1−12.
(60) Carbonell, P., et al. (2018) An automated Design-Build-Test-Learn pipeline for enhanced microbial production of fine chemicals. *Commun. Biol. 1*, 1−10.
(61) Hecht, A., Filliben, J., Munro, S. A., and Salit, M. (2018) A minimum information standard for reproducing bench-scale bacterial cell growth and productivity. *Commun. Biol. 1*, 1−9.
(62) Standards in synthetic biology: gaps, challenges and opportunities. https://standardsinsynbio.eu/wp-content/uploads/2020/09/Deliverable-1.1.pdf (Accessed on 2020−12−24).
(63) Qi, L. S., Larson, M. H., Gilbert, L. A., Doudna, J. A., Weissman, J. S., Arkin, A. P., and Lim, W. A. (2013) Repurposing CRISPR as an RNA-guided platform for sequencespecific control of gene expression. *Cell 152*, 1173−1183.
(64) Silverman, A. D., Karim, A. S., and Jewett, M. C. (2020) Cell-free gene expression: an expanded repertoire of applications. *Nat. Rev. Genet. 21*, 151−170.
(65) Kelly, J. R., Rubin, A. J., Davis, J. H., Ajo-Franklin, C. M., Cumbers, J., Czar, M. J., de Mora, K., Glieberman, A. L., Monie, D. D., and Endy, D. (2009) Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng. 3*, 1−13.
(66) Rudge, T. J., Brown, J. R., Federici, F., Dalchau, N., Phillips, A., Ajioka, J. W., and Haseloff, J. (2016) Characterization of Intrinsic Properties of Promoters. *ACS Synth. Biol. 5*, 89−98.
(67) Exley, K., Reynolds, C. R., Suckling, L., Chee, S. M., Tsipa, A., Freemont, P. S., McClymont, D., and Kitney, R. I. (2019) Utilising datasheets for the informed automated design and build of a synthetic metabolic pathway. *J. Biol. Eng. 13*, 1−10.
(68) Espah Borujeni, A., Zhang, J., Doosthosseini, H., Nielsen, A. A., and Voigt, C. A. (2020) Genetic circuit characterization by inferring RNA polymerase movement and ribosome usage. *Nat. Commun. 11*, 1−18.
(69) Meng, F., and Ellis, T. (2020) The second decade of synthetic biology: 2010−2020. *Nat. Commun. 11*, 1−4.
(70) Lopez, R., Gayoso, A., and Yosef, N. (2020) Enhancing scientific discoveries in molecular biology with deep generative models. *Mol. Syst. Biol. 16*, 1−21.
(71) Hillson, N., et al. (2019) Building a global alliance of biofoundries. *Nat. Commun. 10*, 1038−1041.
(72) Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C. (2016) *Jupyter Notebooks—A Publishing Format for Reproducible Computational Workflows*; ELPUB2016, Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany.
(73) Fedosejev, A. (2016) *React.Js Essentials*; Packt Publishing.
(74) Bostock, M., Ogievetsky, V., and Heer, J. (2011) D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 2301−2309.
(75) Merkel, D. (2014) Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment.
(76) Engler, C., Kandzia, R., and Marillonnet, S. (2008) A one pot, one step, precision cloning method with high throughput capability. *PLoS One 3*, 1−7.
(77) Gibson, D. G., Young, L., Chuang, R. Y., Venter, J. C., Hutchison, C. A., and Smith, H. O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat. Methods 6*, 343−345.

https://doi.org/10.1021/acssynbio.0c00554
ACS Synth. Biol. 2021, 10, 183−191